(19) JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11) Publication number: **08030569 A**

(43) Date of publication of application: **02.02.96**

(51) Int. Cl **G06F 15/163**
G06F 9/45

(21) Application number: **06167905**

(22) Date of filing: **20.07.94**

(71) Applicant: **FUJITSU LTD**

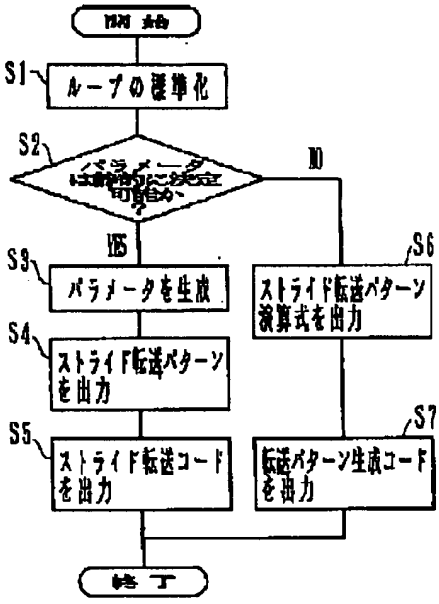
(72) Inventor: **DOI SANEHISA**
SHINDO TATSUYA

(54) METHOD AND DEVICE FOR TRANSFERRING DATA

(57) Abstract:

PURPOSE: To provide data transferring method and device capable of improving the performance of a distributed memory type parallel computer or the like by quickly transferring discontinuous data by hardware.

CONSTITUTION: Loop standardization for the pattern of irregular and discontinuous data described by a DO loop of parallel processing language is executed (step S1). When a parameter for specifying data to be transferred can be statically determined, a compiler outputs a stride transfer pattern supported by hardware (step S4) to execute stride transfer by the hardware. When the parameter can not be statically determined, the compiler outputs an operation expression and a transfer pattern generating code (steps S6, S7) and a library driven at the time of execution generates a stride transfer pattern to execute stride transfer by the hardware.



(19)日本国特許庁（ＪＰ）　(12)**公開特許公報（Ａ）**　(11)特許出願公開番号

特開平8－30569

(43)公開日　平成８年(1996)２月２日

(51)Int.Cl.[6]	識別記号	庁内整理番号	Ｆ Ｉ		技術表示箇所
Ｇ０６Ｆ　15/163					
9/45					
			Ｇ０６Ｆ　15/ 16	３２０　Ｚ	
		7737－5Ｂ	9/ 44	３２２　Ｆ	

審査請求　未請求　請求項の数14　ＯＬ　（全 13 頁）

(21)出願番号　特願平6－167905

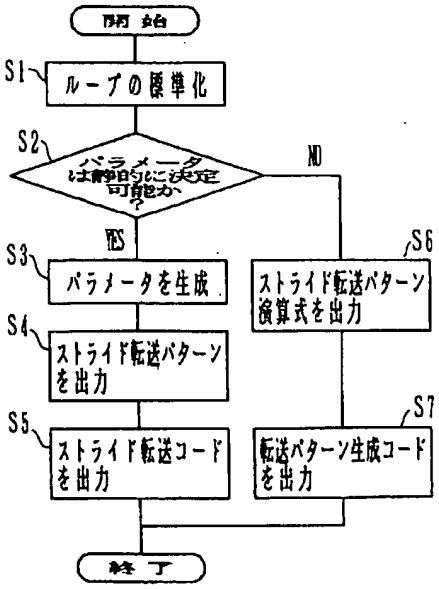
(22)出願日　平成６年(1994)７月20日

(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中1015番地
(72)発明者　土肥　実久
神奈川県川崎市中原区上小田中1015番地
富士通株式会社内
(72)発明者　進藤　達也
神奈川県川崎市中原区上小田中1015番地
富士通株式会社内
(74)代理人　弁理士　大菅　義之　（外１名）

(54)【発明の名称】　データ転送方法とその装置

(57)【要約】
【目的】　ハードウェアを用いて不連続データを高速に転送し、分散メモリ型並列計算機システム等のパフォーマンスを向上させるデータ転送方法とその装置を提供する。
【構成】　並列処理言語のdoループにより記述される規則的な不連続データのパターンを対象とし、ループの標準化を行う（ステップＳ１）。転送対象データを指定するパラメータが静的に決定できる場合は、ハードウェアのサポートするストライド転送パターンをコンパイラが出力し（ステップＳ４）、ハードウェアによるストライド転送を行う。パラメータが静的に決定できない場合は、コンパイラは演算式と転送パターン生成コードを出力し（ステップＳ６、Ｓ７）、実行時のライブラリがストライド転送パターンを生成して、ハードウェアによるストライド転送を行う。

コンパイラによる処理のフローチャート（その１）



1

【特許請求の範囲】

【請求項１】　データ転送を伴う処理を記述し、
記述された該処理から転送対象データを指定するパラメータを生成し、
前記データ転送を実行するハードウェアがサポートする転送パターンを求める演算式を生成し、
生成された前記パラメータと前記演算式とを用いて前記転送パターンを生成し、
得られた該転送パターンを前記ハードウェアに与えて、前記転送対象データの転送を行わせることを特徴とするデータ転送方法。

【請求項２】　前記処理のコンパイル時に、前記パラメータと前記演算式を生成して前記転送パターンを求め、前記ハードウェアを用いたデータ転送を指示するコードを含むプログラムを生成し、
該プログラムの実行時に、前記転送パターンを前記ハードウェアに与えて、前記転送対象データの転送を行わせることを特徴とする請求項１記載のデータ転送方法。

【請求項３】　前記処理のコンパイル時に、前記演算式を生成して、前記転送パターンの生成を指示するコードを含むプログラムを生成し、
該プログラムの実行時に、前記パラメータを生成して、前記演算式により前記転送パターンを求め、該転送パターンを前記ハードウェアに与えて、前記転送対象データの転送を行わせることを特徴とする請求項１記載のデータ転送方法。

【請求項４】　前記処理のコンパイル時に前記パラメータを生成し、前記転送パターンの生成を指示するコードを含むプログラムを生成し、
該プログラムの実行時に、前記パラメータと前記演算式とを用いて前記転送パターンを求め、該転送パターンを前記ハードウェアに与えて、前記転送対象データの転送を行わせることを特徴とする請求項１記載のデータ転送方法。

【請求項５】　分散メモリ型並列計算機システムのための並列処理言語により、前記処理を一括データ転送として記述することを特徴とする請求項１記載のデータ転送方法。

【請求項６】　インデックスを用いたループにより前記データ転送を伴う処理を記述し、
該ループの開始に対応するインデックスと、該ループの終了に対応するインデックスと、該ループにおけるインデックスの間隔のうち少なくとも１つを用いて、前記パラメータを生成することを特徴とする請求項１記載のデータ転送方法。

【請求項７】　前記ハードウェアは、一定の間隔をおいて規則的に格納されたデータのストライド転送を行う機構を持ち、
該ストライド転送を行う機構に対応した前記転送パターンを自動的に生成し、

2

前記転送対象データのストライド転送を前記ハードウェアに行わせることを特徴とする請求項１記載のデータ転送方法。

【請求項８】　前記転送対象データに含まれる転送単位の大きさと、転送単位の数と、最初の転送単位の位置と、２つの転送単位の間隔のうち、少なくも１つを前記転送パターンとして求め、前記ストライド転送を行う機構に入力することを特徴とする請求項６記載のデータ転送方法。

【請求項９】　複数のプロセッサのそれぞれに対応する複数のメモリを用意し、
前記転送対象データの最初の転送単位と最後の転送単位とを互いに異なる前記メモリに格納し、
前記最初の転送単位を有するメモリに対応するプロセッサの識別子と前記転送パターンとを生成することを特徴とする請求項１記載のデータ転送方法。

【請求項１０】　複数のプロセッサのそれぞれに対応する複数のメモリを用意し、
前記転送対象データの最初の転送単位と最後の転送単位とを互いに異なる前記メモリに格納し、
前記最後の転送単位を有するメモリに対応するプロセッサの識別子と前記転送パターンとを生成することを特徴とする請求項１記載のデータ転送方法。

【請求項１１】　複数のプロセッサのそれぞれに対応する複数のメモリを用意し、
前記転送対象データの最初の転送単位と最後の転送単位とを互いに異なる前記メモリに格納し、
前記最初の転送単位と最後の転送単位を共に含まないメモリに対応するプロセッサの識別子と前記転送パターンとを生成することを特徴とする請求項１記載のデータ転送方法。

【請求項１２】　複数のプロセッサのそれぞれに対応する複数のメモリを用意し、
前記転送対象データを前記複数のメモリのうちの１つに格納し、
該１つのメモリに対応するプロセッサの識別子と前記転送パターンとを生成することを特徴とする請求項１記載のデータ転送方法。

【請求項１３】　複数のプロセッサのそれぞれに対応する複数のメモリを用意し、
前記転送対象データを２つ以上の前記メモリにサイクリックに分割して格納し、
前記転送対象データが格納された前記２つ以上のメモリのうちの１つに対応するプロセッサの識別子と、該１つのメモリに格納された前記転送対象データに関する前記転送パターンとを生成することを特徴とする請求項１記載のデータ転送方法。

【請求項１４】　ハードウェアによりデータを転送する方式であって、
プログラムに記述されたデータ転送を伴う処理から転送

対象データを指定するパラメータを生成し、前記ハードウェアがサポートする転送パターンを求める演算式を生成する手段と、

前記パラメータと前記演算式とを用いて前記転送パターンを生成し、該転送パターンを前記ハードウェアに与えて、前記転送対象データの転送を行わせる手段とを有することを特徴とするデータ転送方式。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は情報処理装置におけるデータ転送に係り、さらに詳しくは、分散メモリ型並列計算機システムにおいて不連続に格納されたデータの転送方法とその装置に関する。

【０００２】

【従来の技術】近年、計算機の計算性能向上に対する要求は日増しに高くなり、この要求に答えるために複数のプロセッサが並列に処理を行う並列計算機が作られるようになってきた。初期においては複数のプロセッサがメモリを共有する共有メモリ型並列計算機が主流であったが、今日ではより並列性の高いプログラムに対して多くのプロセッサを実装するために、各プロセッサがそれぞれ別のメモリを持つ分散メモリ型並列計算機が作られるようになってきている。

【０００３】このような分散メモリ型並列計算機では、１種類のデータを格納した変数を複数のプロセッサのメモリに分散して持たせ、それぞれのプロセッサが極力自己のメモリ内に持っているデータに関してのみ計算を行うようプログラミングをして高速化を図っている（プロセッサローカル）。

【０００４】しかし実際には、一般的なプログラムの場合に全ての演算をプロセッサローカルで計算することはできず、他のプロセッサの持っているデータを必要とすることがある。このとき、必要なデータを持っているプロセッサからそのデータを必要としているプロセッサへの通信が必要だが、この通信に要する時間は本来の計算処理に対するオーバヘッドとなる。

【０００５】一般に分散メモリ型並列計算機の通信には、１回毎に送信先を指定したり、ネットワークとの接続を指定したりする処理等の立ち上がりオーバヘッドが存在する。このため、通信により転送されるデータの総量が等しければ、その通信回数が少ないほど全体の通信時間を短縮することができる。そこで、従来から１回の通信でより多くのデータを送れるように、メモリ上の連続領域内のデータはまとめて転送し、不連続な幾つかの領域にまたがるデータ（不連続データ）はソフトウェアによりパッキングして転送していた。

【０００６】

【発明が解決しようとする課題】しかしながら、従来の不連続データ転送方法では、送信側でデータをパッキングしたり、受信側でパッキングされたデータを展開したりする処理のオーバヘッドが必要であり、高速な通信を行うことができないという問題がある。

【０００７】また、トランスポーズ転送に代表される規則的なパターンで表現されるような不連続データを転送するハードウェアであるストライドデータ転送機構を持っていても、プログラマにより直接その機構を用いた転送処理が記述されなければ使用することはできなかった。

【０００８】このため、このような規則的なパターンの通信であっても、一般には不規則な不連続データの通信と同様に、送信側でデータのパッキングを行い、受信側でパッキングされたデータを展開するような転送を指示するコードをコンパイラが生成していた。したがって、ストライドデータ転送機構が十分に活用されず、規則的なパターンの通信時間が短縮されていなかった。

【０００９】本発明は、情報処理装置における不連続データの転送を高速に行うデータ転送方法とその装置を提供することを目的とする。さらに詳しくは、ストライドデータ転送機構を用いるための通信コードをコンパイラが出力することにより高速な通信を行い、分散メモリ型並列計算機システム等のパフォーマンスを向上させることを目的とする。

【００１０】

【課題を解決するための手段】本発明は、ループなどにより規則的なパターンとして表現される不連続データをストライドデータ転送機構などのハードウェアにより転送する方法とその装置である。

【００１１】図１は、本発明のデータ転送方法を示す原理図である。本発明のデータ転送方法においては、まずデータ転送を伴う処理を記述し（ステップＳＴ１）、記述された処理から転送対象データを指定するパラメータを生成し（ステップＳＴ２）、ハードウェアがサポートする転送パターンを求める演算式を生成する（ステップＳＴ３）。

【００１２】そして、生成されたパラメータと演算式とを用いて転送パターンを生成し、得られた転送パターンをハードウェアに与えて、転送対象データの転送を行わせる（ステップＳＴ４）。

【００１３】転送対象データを指定するパラメータは、例えばループの開始インデックス、終了インデックス、連続する２つのインデックスの間隔等に対応する値であり、転送パターンは、例えばストライドデータ転送機構がサポートする不連続データの転送単位の大きさ、転送単位の数、最初の転送単位の位置、２つの転送単位の間隔等により指定される。

【００１４】本発明では、データ転送を伴う処理のコンパイル時にコンパイラが転送パターンを生成するか、またはプログラムの実行時に自動的に転送パターンを生成させるコードを生成する。後者の場合には、そのコードの実行時にライブラリ等が必要な転送パターンを生成す

る。

【0015】

【作用】コンパイラまたはプログラム実行時のライブラリ等が、転送対象データを指定するパラメータと転送パターンを求める演算式とを自動的に生成するので、個々の転送対象データに応じた転送パターンを計算することができる。

【0016】また、転送パターンを自動的に生成して、ストライドデータ転送機構などのハードウェアに与えるので、プログラム実行時にハードウェアによる不連続データの転送が可能になり、ハードウェアの利用率が向上する。

【0017】さらに、不連続領域のデータをパッキングした後に展開するという処理が不要になり、高速なデータ転送が行われる。したがって、情報処理装置のパフォーマンスが大きく向上する。

【0018】

【実施例】以下図面を参照しながら、本発明の実施例について説明する。図2は、本発明のデータ転送方法を用いる分散メモリ型並列計算機システムの構成図である。図2において、N個のプロセッサ1－1、1－2、・・・、1－Nはそれぞれに接続されたメモリ2－1、2－2、・・・、2－Nを有し、これらのN個のメモリはネットワーク3により接続されている。

【0019】各メモリ2－1、2－2、・・・、2－Nの一部はそれぞれローカルメモリ5－1、5－2、・・・、5－Nとして各プロセッサ1－1、1－2、・・・、1－Nが用いるローカル変数等を格納する。また、メモリ2－1、2－2、・・・、2－Nの残りの領域はグローバルメモリ4としてプロセッサ1－1、1－2、・・・、1－Nが共有するグローバル変数等を格納する。

【0020】N個のプロセッサはグローバルメモリ4と自己の持つローカルメモリにアクセスしながら並列に処理を行い、他のプロセッサの有するデータが必要になるとネットワーク3を介して通信を行う。

【0021】本実施例では、多次元のデータパターンのうち、各次元についてループインデックスを持ち、お互いに独立なdoループによって記述されるものを対象とする。そして、メモリに格納されたこのデータを処理するソースプログラムをコンパイルするとき、またはそのプログラムを実行したときに、配列の各次元について、メモリ上でのループの開始位置を指定するパラメータと、ループの終了位置を指定するパラメータと、転送すべき配列要素の間隔を指定するパラメータと、対象となる配列データの格納パターンにしたがって解析を行い、ストライドデータ転送のための転送パターン（ストライド転送パターン）を生成する。

【0022】図3は、本実施例における1次元のデータのストライド転送パターンを示している。本実施例におけるストライドデータ転送では、通信に関わる2つのプロセッサ（ローカルとリモート）のメモリ上で共通の大きさを持つ連続領域を転送単位（斜線部分）とし、この転送単位の大きさをSizeとする。また、ローカル、リモートのそれぞれのメモリ上で、1つの転送単位が現れてから次の転送単位が現れるまでの大きさをStrideとし、転送単位の数mをCount とする。さらに、ストライド転送パターンのメモリ上での開始アドレスをStart として、これらの4つのパラメータStart 、Size、Stride、Count によりストライド転送パターンを指定する。

【0023】複数次元についてデータが規則的に配置された多次元のストライド転送パターンの場合には、その次元の1つ下の次元のストライド転送パターンを、その次元における転送単位として考える。例えば、2次元のストライド転送パターンにおける転送単位は1次元のストライド転送パターンであり、この1次元のストライド転送パターンの数が2次元のストライド転送パターンにおけるCount となる。

【0024】Fortranに代表される既存言語における配列変数間のデータ転送のパターンは多くの場合、各次元毎に独立なループインデックスを持ちお互いに独立なdoループによって記述され得るため、このパターンはこれらのループにより規則的に表現できる。また、各次元に対応するストライド転送パターンは互いに直交しているため、各次元毎にStart 、Size、Stride、Count で示されるストライド転送パターンを求めれば、1次元のストライド転送パターンを任意のn次元にまで拡張することもできる。

【0025】図4は、 Fortranにおけるdoループの一例を示している。図4（a）のdoループは、インデックスiの1から5までの各値について、2次元の配列変数a（1，2＊i）の値を1次元の配列変数b（i＋1）に書き込む処理を表す。もし、配列変数a（1，2＊i）とb（i＋1）が同じプロセッサのメモリ内になければ、ネットワーク3を介したデータ転送が必要になる。図4（a）に記述された処理は、並列処理言語の表記による一括データ転送に相当する。

【0026】図4（b）、（c）のdoループは、それぞれ配列変数a（j，k）、b（h）を有するプロセッサのプログラムにおいて、図4（a）のdoループを標準化した結果を示している。ループの標準化はコンパイラにより行われる。図4（b）、（c）のインデックス i_g 、i_l はそれぞれ配列変数a（j，k）、b（h）のインデックスk、hに対応しており、インデックスの添字$_g$ 、$_l$ はそれぞれ図5におけるグローバル変数、ローカル変数に対応している。

【0027】図4（b）において、i_g＝2，10，2はdoループの開始位置がi_g＝2、doループの終了位置がi_g＝10、doループのインデックスの間隔が2であることを表す。また、図4（c）において、i_l＝2，

6はdoループの開始位置がi_L＝2、doループの終了位置がi_L＝6、doループのインデックスの間隔が1であることを表す。doループのインデックスの間隔が1であるときは、このパラメータは省略される。

【0028】図4（b）のdoループの中の四角は、メモリから読み出されたa（1，i_c）の値を送信するために一時格納する記憶域、または直接通信の場合はネットワーク3を表す。図4（c）のdoループの中の四角は、受信したa（1，i_c）の値を一時格納する記憶域、または直接通信の場合はネットワーク3を表す。

【0029】本実施例では、ストライド転送がグローバル変数とローカル変数の2変数間で行われるものとする。ここで、ローカル変数はプロセッサが有するローカルメモリを指すローカル空間にある変数であり、グローバル変数はグローバルメモリ4を指すグローバル空間にある変数である。グローバル変数ではインデックスを指定するとそのオーナーであるプロセッサとメモリ上の格納アドレスは一意に決定される。この条件の下でグローバル変数について、転送対象となる次元の分割形状に応じたストライド転送パターンを生成する。

【0030】図5は、図4の配列変数a（j，k）、b（h）の格納構造の一例とそれらの間のストライド転送を示している。図5において、a（j，k）（j＝1，10、k＝1，10）はグローバル変数としてグローバルメモリ4に格納されており、b（h）（h＝1，10）はローカル変数としてローカルメモリ5－1～5－Nのいずれかに格納されている。

【0031】この場合、変数a（j，k）の最初のインデックスjはメモリ上でデータが連続して格納される方向（1次元方向）のインデックスで、2番目のインデックスkはこれに直交する方向（2次元方向）のインデックスである。また、変数b（h）のインデックスhはメモリ上の連続方向を表す。

【0032】変数a（j，k）から変数b（h）へ転送すべき配列要素は、i＝1、2、3、4、5に対応して斜線で示されたa（1，2）、a（1，4）、a（1，6）、a（1，8）、a（1，10）の5つである。これらのデータは、それぞれb（2）、b（3）、b（4）、b（5）、b（6）に格納される。

【0033】したがって、図5の1つの枡目に対応するデータが転送単位となり、そのメモリ上の大きさがSizeで、Countは5である。また、グローバル変数におけるStrideは、メモリ上で例えば転送単位a（1，2）が現れてから次の転送単位a（1，4）が現れるまでの大きさであり、ここではSizeの20倍に相当する。グローバル変数におけるStartは転送単位a（1，2）の先頭アドレスである。

【0034】一方、ローカル変数におけるStrideは、例えばb（2）からb（3）までの大きさでSizeに一致する。ローカル変数におけるStartはb（2）の先頭アドレスである。

【0035】次に、グローバル変数の転送対象となる次元がそれぞれ異なるプロセッサに属するメモリにブロック分割されている場合のストライド転送パターンの導出方法を説明する。

【0036】図6は、グローバル変数a（j，k）がブロック分割されている場合のストライド転送を示している。グローバル変数がブロック分割されているときは、アクセス対象のデータ領域を持つプロセッサは分割境界で切り替わる。

【0037】図6において、グローバル変数a（j，k）はプロセッサ1－1、1－2、1－3によりインデックスkの方向（2次元方向）に3つのブロックに分割されている。k＝1～4の要素はプロセッサ1－1のメモリ2－1に格納され、k＝5～8の要素はプロセッサ1－2のメモリ2－2に格納され、k＝9、10の要素はプロセッサ1－3のメモリ2－3に格納されている。そして、k＝4とk＝5、k＝8とk＝9の間がそれぞれブロックの分割境界となっている。

【0038】グローバル変数がブロック分割されているとき、そのローカル変数へのストライド転送パターンのパラメータは、一般に次式により求められる。

【0039】

【数1】

$$Size = elementSize \quad \cdots (1)$$

$$LACK(m,d) = \begin{cases} MOD(m,d) = 0 \text{のとき } 0, & \cdots (2-1) \\ MOD(m,d) \neq 0 \text{のとき } d - MOD(m,d) & \cdots (2-2) \end{cases}$$

$$LACK_e = LACK(blockStart_e - loopStart_e, loopStride_e) \quad \cdots (2-3)$$

$$Start_e = \begin{cases} loopStart_e & \cdots (2-4) \\ blockStart_e + LACK_e & \cdots (2-5) \end{cases}$$

$$Stride_e = loopStride_e \times blocksize_e \quad \cdots (3)$$

$$Stride_L = loopStride_L \times blocksize_L \quad \cdots (4)$$

$$Width_e = \begin{cases} loopEnd_e - Start_e + 1 & \cdots (5-1) \\ blockEnd_e - Start_e + 1 & \cdots (5-2) \end{cases}$$

$$Count = \frac{Width_e + loopStride_e - 1}{loopStride_e} \quad \cdots (6)$$

【００４０】（１）式において、elementSize はグローバル変数とローカル変数の双方におけるデータの格納単位の大きさを表し、これがそのままストライド転送パターンにおけるSizeとなる。例えば図６においてはSize＝１である。

【００４１】整数m、dに対するＬＡＣＫ（m，d）を定義する（２－１）、（２－２）式において、ＭＯＤ（m，d）はmをdで割ったときの整数剰余を表す。ＭＯＤ（m，d）＝０のときは（２－１）式によりＬＡＣＫ（m，d）＝０であり、ＭＯＤ（m，d）≠０のときは（２－２）式によりＬＡＣＫ（m，d）＝d－ＭＯＤ（m，d）である。（２－１）、（２－２）式を用いて、$LACK_e$ は（２－３）式により定義される。

【００４２】（２－３）、（２－４）、（２－５）式において、 $loopStart_e$ はグローバル変数におけるdoループの開始位置を表し、$blockStart_e$ は対応するプロセッサに割り当てられたブロックの開始位置を表す。ここで開始位置とは、メモリ上のアドレスに対応する変数のインデックスを意味する。$loopStride_e$ はグローバル変数におけるdoループのインデックスの間隔を表す。

【００４３】グローバル変数におけるアドレス Startに対応する開始位置 $Start_e$ を求めるとき、対応するプロセッサのメモリ内に $loopStart_e$ があれば（２－４）式を用い、 $loopStart_e$ が他のプロセッサのメモリ内にあれば（２－５）式を用いる。

【００４４】図６においては、図４（ｂ）に示されるように $loopStart_e$ ＝２であり、 $loopStart_e$ はメモリ２－１内にあるので、プロセッサ１－１については（２－４）式により $Start_e$ ＝２となる。

【００４５】また、図４（ｂ）に示されるように$loopStride_e$ ＝２であり、プロセッサ１－２については、図６からblockStart$_e$ ＝５であるので、（２－３）式により$LACK_e$ ＝ＬＡＣＫ（５－２，２）＝ＬＡＣＫ（３，２）となる。ここで、ＭＯＤ（３，２）＝１≠０であるから、（２－２）式によりＬＡＣＫ（３，２）＝２－１＝１となる。したがって、プロセッサ１－２については（２－５）式により $Start_e$ ＝５＋１＝６となる。

【００４６】プロセッサ１－２については図６からblockStart$_e$ ＝９であるので、同様に（２－５）式により $Start_e$ ＝９＋ＬＡＣＫ（９－２，２）＝１０となる。逆に $Start_e$ が決まると、対応するプロセッサの識別子も特定される。

【００４７】（３）式において、 $blocksize_e$ は隣の要素までのメモリ上の大きさを表す。グローバル変数におけるStrideを表す$Stride_e$ は、$loopStride_e$ と $blocksize_e$の積により求められる。ここでは、 $blocksize_e$ はメモリ上でａ（１，ｋ）からａ（１０，ｋ）までの要素数１０に等しいので、（３）式により$Stride_e$ ＝２０となる。

【００４８】（４）式において、$loopStride_L$ はローカル変数におけるdoループのインデックスの間隔を表し、 $blocksize_L$ は隣の要素までのメモリ上の大きさを表す。ローカル変数におけるStrideを表す$Stride_L$ は、これらの積により求められる。ここでは、図４（ｃ）に示されるように$loopStride_L$ ＝１であり、図６から明らかなように $blocksize_L$ ＝１であるので、（４）式により$Stride_L$ ＝１となる。

【００４９】（５－１）式において、 $loopEnd_e$ はグローバル変数におけるdoループの終了位置を表し、（５－２）式において、$blockEnd_e$ は対応するブロックの終了位置を表す。ここでの終了位置も開始位置と同様に、メモリ上のアドレスに対応する変数のインデックスを意味

する。これらの式において、 $Width_c$ は対応するプロセッサに割り当てられたグローバル変数の転送データが存在する範囲の要素数を表している。ここでは、図4（b）に示されるように $loopEnd_c$ ＝10である。

【0050】グローバル変数におけるdoループの終了位置が対応するブロック内にあるときは（5－1）式により、終了位置が他のブロック内にあるときは（5－2）式により、 $Width_c$ が求められる。

【0051】ここでは、プロセッサ1－1については、 $loopEnd_c$ が対応するブロック内にはなく、 $Start_c$ ＝2、 $blockEnd_c$ ＝4なので、（5－2）式により $Width_c$ ＝3となる。プロセッサ1－2についても $loopEnd_c$ がなく、 $Start_c$ ＝6、 $blockEnd_c$ ＝8なので、（5－2）式により $Width_c$ ＝3となる。また、プロセッサ1－3については、 $loopEnd_c$ ＝10が対応するブロック内にあり、 $Start_c$ ＝10なので、（5－1）式により $Width_c$ ＝1となる。

【0052】ストライド転送パターンにおけるCountは、（6）式により求められる。ここでは、プロセッサ1－1、1－2については $Width_c$ ＝3、 $loopStride_c$ ＝2であるので、（6）式によりCount＝2となる。また、プロセッサ1－3については $Width_c$ ＝1であるから、（6）式によりCount＝1となる。

【0053】このようにして、プロセッサのメモリにdoループの開始位置のみがある場合（プロセッサ1－1）と、終了位置のみがある場合（プロセッサ1－3）と、どちらもない場合（プロセッサ1－2）とについて、ストライド転送パターンを求めることができる。

【0054】グローバル変数の転送対象となる次元がブロックに分割されていない場合は、転送対象となる全ての要素が同じプロセッサのメモリにあるので、doループの開始位置と終了位置の両方が対応するブロック内にあるものとして扱うことができる。

【0055】例えば図5のグローバル変数a（j，k）がこの場合に相当し、（2－4）式により $Start_c$ ＝ $loopStart_c$ ＝2となり、 $loopEnd_c$ ＝10なので（5－1）により $Width_c$ ＝9となる。また $loopStride_c$ ＝2なので、（6）式によりCount＝5となる。Sizeと $Stride_c$ は（1）式と（3）式により求められ、それぞれ1と20になる。

【0056】尚、ローカル変数は常に1つのプロセッサのメモリに格納されるので、そのストライド転送パターンの開始位置 $Start_L$ は、グローバル変数の分割形態に依らずにローカル変数のdoループの開始位置に一致する。例えば、図5、図6の場合には $Start_L$ ＝2である。

【0057】図7は、グローバル変数a（j，k）がサイクリック分割されている場合のストライド転送を示している。グローバル変数の転送対象となる次元がサイクリック分割されている場合、隣合う要素はそのオーナーが別プロセッサとなる。

【0058】図7においては、グローバル変数a（j，k）は3台のプロセッサ1－1、1－2、1－3によりインデックスkの方向にサイクリックに分割されている。k＝1、4、7、10の要素はプロセッサ1－1のメモリ2－1に格納され、k＝2、5、8の要素はプロセッサ1－2のメモリ2－2に格納され、k＝3、6、9の要素はプロセッサ1－3のメモリ2－3に格納されている。そして、k＝3とk＝4、k＝6とk＝7、k＝9とk＝10の間がそれぞれ分割周期の境界となっている。

【0059】グローバル変数がサイクリック分割されているとき、そのローカル変数へのストライド転送パターンのパラメータは、一般に次式により求められる。

【0060】

【数2】

$$Size = elementSize \quad \cdots (7)$$

$$GCD_{cs} = GCD(procNum_c, loopStride_c) \quad \cdots (8)$$

$$LCM_{cs} = LCM(procNum_c, loopStride_c) \quad \cdots (9)$$

$$offset = \left\{ n \mid 0 \leq n \leq \frac{procNum_c}{GCD_{cs}} - 1 \right\} \quad \cdots (10)$$

$$Start_c = loopStart_c + offset \times loopStride_c \quad \cdots (11)$$

$$Stride_c = \frac{LCM_{cs} \times blocksize_c}{procNum_c} \quad \cdots (12)$$

$$Stride_L = \frac{procNum_c \times loopStride_L \times blocksize_L}{GCD_{cs}} \quad \cdots (13)$$

$$Count = \frac{loopEnd_c - Start_c + LCM_{cs}}{LCM_{cs}} \quad \cdots (14)$$

【００６１】（７）～（１４）式において、（１）～（６）式で用いたものと同じ記号は（１）～（６）式と同様の意味を持つ。また、Sizeを求める（７）式は（１）式と同じである。例えば図７の場合は（７）式によりSize＝１となる。

【００６２】（８）、（９）式において、$procNum_c$ はグローバル変数をサイクリック分割しているプロセッサの数を表し、ＧＣＤ（$procNum_c$，$loopStride_c$）は$procNum_c$と$loopStride_c$の最大公約数、ＬＣＭ（$procNum_c$，$loopStride_c$）は$procNum_c$と$loopStride_c$の最小公倍数を表す。GCD_{cs}、LCM_{cs}は（８）、（９）式により定義される。

【００６３】図７の場合は、$procNum_c$＝３、$loopStride_c$＝２であるので、（８）、（９）式によりGCD_{cs}＝１、LCM_{cs}＝６となる。（１０）式のｎは整数を表し、offsetは｛　｝内の不等式を満たすｎの値を表す。

【００６４】ここでは、（１０）式の｛　｝内の不等式は$0 \leq n \leq 2$となるので、offset＝０、１、２となる。このとき、$loopStart_c$＝２なので、（１１）式によりoffsetの各値に対応して$Start_c$＝２、４、６となる。これらの$Start_c$の値は、それぞれメモリ２－２、２－１、２－３に格納されているストライド転送パターンの開始位置に対応している。したがって、$Start_c$が決まれば、対応するプロセッサの識別子も特定される。

【００６５】また、$Stride_c$と$Stride_L$は（１２）式と（１３）式により求められ、それぞれ２０と３になる。（１２）式の右辺のLCM_{cs}／$procNum_c$は、実際のプロセッサのメモリ上でのdoループのインデックスの間隔を表している。$Stride_c$の値は図６のブロック分割の場合と同じであるが、$Stride_L$の値は異なる。これは、例えばメモリ２－２において、転送単位ａ（１，２）の位置から次の転送単位ａ（１，８）の位置までの大きさが２０であるが、変数ｂ（ｈ）においては、これらの転送単位に対応する格納位置がｂ（２）とｂ（５）であって、必ずしも隣接していないことに対応している。

【００６６】（１４）式においては、右辺の商をCountの値とする。これは、（$loopEnd_c$－$Start_c$＋１）／LCM_{cs}が割り切れない時に、その商に１を加算した値をCountとすることに相当する。図７の場合には、$loopEnd_c$＝１０、$Start_c$＝２、４、６、LCM_{cs}＝６なので、（１４）式により$Start_c$の各値に対応してCount＝２、２、１となる。これらのCountの値は、それぞれメモリ２－２、２－１、２－３に格納されている転送単位の数に対応している。

【００６７】図８は、９台のプロセッサによりサイクリック分割されたグローバル変数を示している。図８において、グローバル変数ａ（ｊ，ｋ）（ｊ＝１，１０、ｋ＝１，２０）は、プロセッサ１－１～１－９のメモリ２－１～２－９にサイクリックに分割されて格納されており、ｋ＝９とｋ＝１０、ｋ＝１８とｋ＝１９の間がそれぞれ分割周期の境界となっている。ストライド転送パターンの転送単位は斜線で示された５つであり、メモリ２－２、２－５、２－８に分かれて格納されている。

【００６８】この場合、（７）式によりSize＝１であり、例えば転送単位ａ（１，２）とａ（１，５）の間隔に対応して$loopStride_c$＝３である。また、$procNum_c$＝９なので、（８）、（９）式によりGCD_{cs}＝３、LCM_{cs}＝９となる。

【００６９】このとき、（１０）式よりoffset＝０、１、２となり、$loopStart_c$＝２なので、（１１）式によりoffsetの各値に対応して$Start_c$＝２、５、８となる。これらの$Start_c$の値は、それぞれメモリ２－２、２－５、２－８に格納されているストライド転送パターンの開始位置に対応している。他のメモリには転送単位

が格納されていない。

【0070】そして、$blocksize_0$ ＝10、$loopEnd_0$ ＝14なので、（12）により$Stride_0$ ＝10となり、（14）式により $Start_0$ の各値に対応してCount ＝2、2、1となる。これらのCount の値は、それぞれメモリ2－2、2－5、2－8に格納されている転送単位の数に対応している。

【0071】次に、図9から図12までを参照しながら、本実施例の並列計算機システムが実行するプログラムのコンパイルを行うコンパイラの処理と、実行時のプログラムおよびライブラリの処理について説明する。図9および図10は本実施例におけるコンパイラの処理を示すフローチャートであり、図11は実行時のプログラムによる処理を示すフローチャートであり、図12は実行時のライブラリの処理を示すフローチャートである。

【0072】本実施例において、ストライド転送パターンをコンパイラが生成する場合はコンパイラは図9の処理を行い、ストライド転送パターンを実行時ライブラリが生成する場合はコンパイラは図10の処理を行う。

【0073】図9において処理が開始されると、コンパイラはまずソースプログラムの中の対象とするループの標準化を行って、プログラマにより記述されたループを変数の格納構造に対応したインデックスにより表されるループに変換する（ステップS1）。このとき、例えば図4（a）のようなdoループは、図4（b）、（c）のようなdoループに変換される。

【0074】次に、ストライド転送パターンを計算するために必要なパラメータの値が静的に決定可能かどうかを判定する（ステップS2）。ここで、必要なパラメータには、標準化されたループに関するループパラメータと変数の格納パターンに関する格納パターンパラメータとが含まれる。

【0075】ループパラメータは、標準化されたループの開始位置（ $loopStart_0$ 、 $loopStart_1$ ）、終了位置（ $loopEnd_0$ 、 $loopEnd_1$ ）、およびインデックスの間隔（$loopStride_0$ 、$loopStride_1$ ）等を指し、格納パターンパラメータは、（1）～（14）式におけるelementSize 、$blockStart_0$ 、$blockEnd_0$ 、 $blocksize_0$ 、$blocksize_1$ 、$blockWidth_0$ 、 $procNum_0$ 等を指す。この格納パターンパラメータは、複数のプロセッサによるグローバル変数の分割形態に関する情報を含んでいる。

【0076】これらのパラメータの値がコンパイル時において決定され、プログラム実行時に変化しない場合は（ステップS2、YES）、ソースプログラムの記述にしたがって各パラメータを生成する（ステップS3）。そして、グローバル変数の分割形態に応じて（1）～（14）式により、生成されたパラメータの値からストライド転送パターンを計算して出力する（ステップS4）。このとき同時に、対応するプロセッサの識別子も出力する。

【0077】次に、プログラム実行時にストライドデータ転送機構を用いたデータ転送を行う通信コードであるストライド転送コードを出力して（ステップS5）、処理を終了する。

【0078】ストライド転送パターンを計算するために必要なパラメータの値がコンパイル時において決定されないとき、あるいはプログラム実行時に動的に変化する場合は（ステップS2、NO）、（1）～（14）に示したストライド転送パターンの一般的な演算式を出力する（ステップS6）。そして、プログラム実行時にストライド転送パターンを生成するコードである転送パターン生成コードを出力して（ステップS7）、処理を終了する。

【0079】一方、図2の並列計算機システムによるプログラム実行時にストライド転送コードが現れると、データ転送に関与するプロセッサはコンパイラの出力したストライド転送パターンをストライドデータ転送機構に与える。これにより、ストライドデータ転送機構がデータ転送を行う。

【0080】また、プログラム実行時に転送パターン生成コードが現れると、プログラムは図11に示す処理を行う。図11において、まず必要なパラメータを生成し（ステップS21）、生成したパラメータを引数としてライブラリを呼び出す（ステップS22）。

【0081】プログラムから呼び出されたライブラリは図12に示す処理を行う。図12において、まず与えられたパラメータとコンパイラの出力したストライド転送パターン演算式を用いてストライド転送パターンを計算する（ステップS31）。このとき同時に、対応するプロセッサの識別子も決定する。そして、得られたストライド転送パターンをストライドデータ転送機構に与える（ステップS32）。これにより、ストライドデータ転送機構がデータ転送を行う。

【0082】コンパイラがストライド転送パターンの生成を行わない場合は、図10の処理を行う。図10において処理が開始されると、コンパイラはまず図9のステップS1と同様のループの標準化を行い（ステップS11）、標準化されたループのループパラメータを出力する（ステップS12）。そして、ストライド転送パターンを生成する転送パターン生成コードを出力して（ステップS13）、処理を終了する。

【0083】一方、転送パターン生成コードが現れると、実行時のプログラムは図11のフローにしたがって、まず格納パターンパラメータを生成する（ステップS21）。次に、生成した格納パターンパラメータとコンパイラの出力したループパラメータとを引数としてライブラリを呼び出す（ステップS22）。

【0084】呼び出されたライブラリは図12のフローにしたがって、与えられたパラメータと（1）～（14）式のストライド転送パターン演算式を用いてストラ

17

イド転送パターンを計算する（ステップＳ３１）。

【００８５】この場合、ライブラリが必要なストライド転送パターン演算式を生成してもよく、また、あらかじめストライド転送パターン演算式をライブラリに組み込んでおいてもよい。また、このとき同時に、対応するプロセッサの識別子も決定する。次に、得られたストライド転送パターンをストライドデータ転送機構に与える（ステップＳ３２）。これにより、ストライドデータ転送機構がデータ転送を行う。

【００８６】

【発明の効果】本発明によれば、コンパイラまたは実行時のライブラリが必要に応じて個々のストライド転送パターンを生成するので、規則的なパターンとして表現される不連続領域のデータを転送するときにストライドデータ転送機構を利用することができる。したがって、不連続領域のデータをパッキングした後に展開するという処理を行うことなく、通信回数を大幅に減少させることが可能となる。

【００８７】これにより、従来プログラマが指定しなければ利用されなかったストライドデータ転送機構の利用率が著しく向上し、高速な通信が実現される。ひいては分散メモリ型並列計算機システム等の情報処理装置のパフォーマンス向上に大きく寄与する。

【図面の簡単な説明】

【図１】本発明の原理図である。

【図２】一実施例における並列計算機システムの構成図である。

18

＊【図３】実施例におけるストライド転送パターンを示す図である。

【図４】実施例におけるdoループの一例を示す図である。

【図５】実施例におけるグローバル変数のストライド転送を示す図である。

【図６】実施例におけるブロック分割されたグローバル変数のストライド転送を示す図である。

【図７】実施例におけるサイクリック分割されたグローバル変数のストライド転送を示す図である。

【図８】実施例における９台のプロセッサによりサイクリック分割されたグローバル変数のストライド転送を示す図である。

【図９】実施例におけるコンパイラによる処理のフローチャート（その１）である。

【図１０】実施例におけるコンパイラによる処理のフローチャート（その２）である。

【図１１】実施例におけるプログラムによる処理のフローチャートである。

【図１２】実施例におけるライブラリによる処理のフローチャートである。

【符号の説明】

１－１、２、・・・、Ｎ　プロセッサ

２－１、２、・・・、Ｎ　メモリ

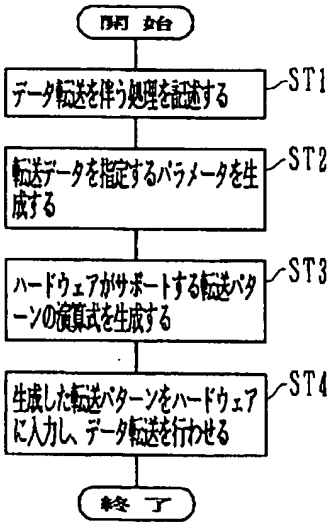
３　ネットワーク

４　グローバルメモリ

５－１、２、・・・、Ｎ　ローカルメモリ

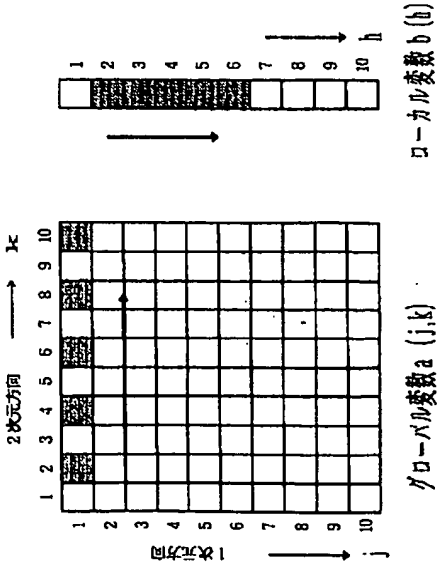
【図１】

本発明の原理図



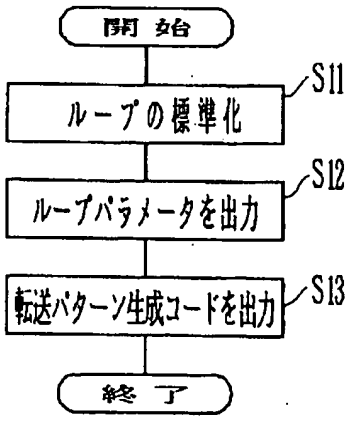
【図５】

グローバル変数のストライド転送を示す図



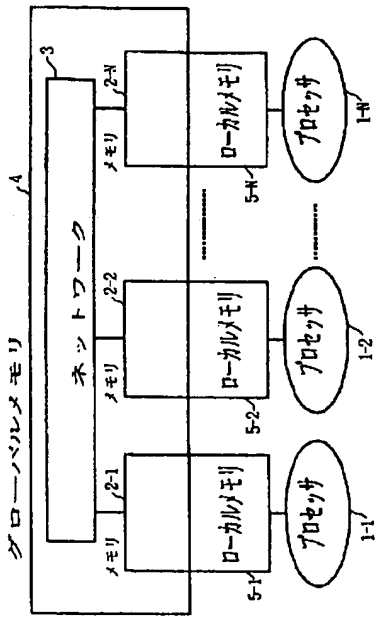
【図１０】

コンパイラによる処理のフローチャート（その２）



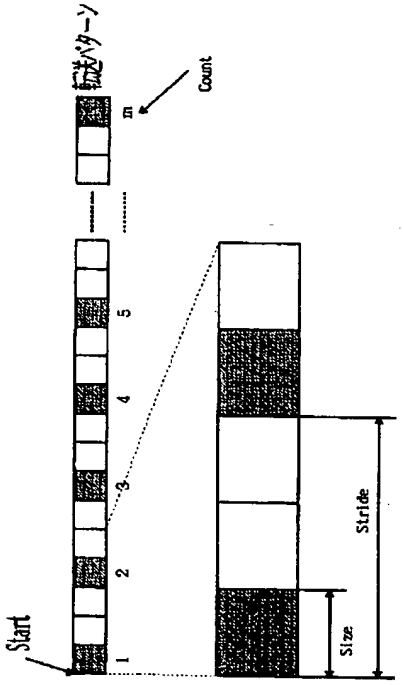
【図2】

一実施例における並列計算機システムの構成図



【図3】

一実施例におけるストライド転送パターンを示す図



【図4】

doループの一例を示す図

(a)

```
do   i=1, 5
     b(i+1)=a(1, 2*i)
end do
```

(b)

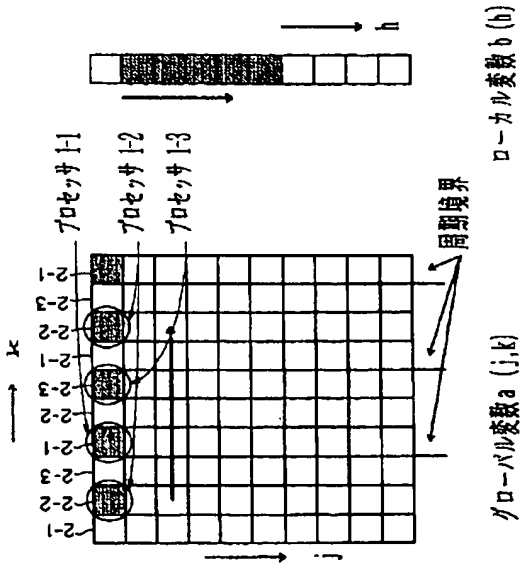
```
do   i_G=2, 10, 2
     [　　] = a (1, i_G)
end do
```

(c)

```
do   i_L=2, 6
     b (i_L) = [　　]
end do
```

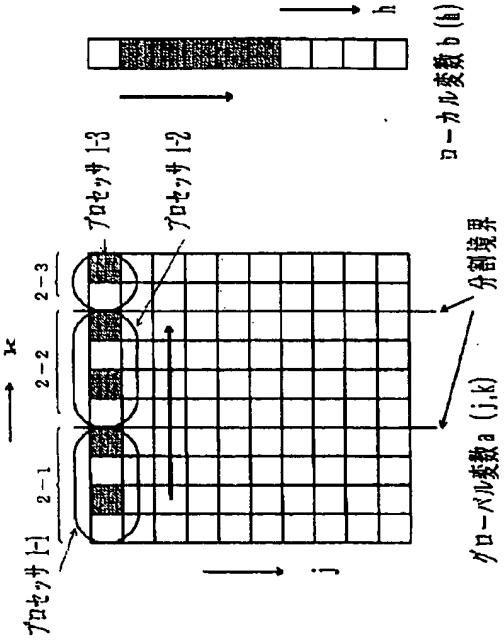
【図7】

サイクリック分割されたグローバル変数のストライド転送を示す図



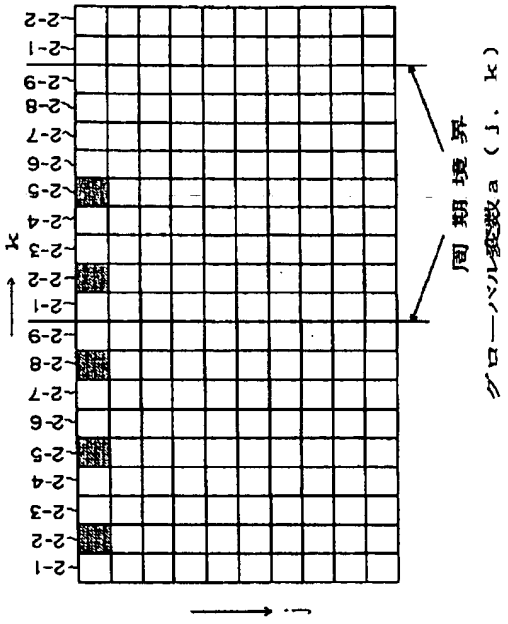
【図6】

ブロック分割されたグローバル変数の
ストライド転送を示す図



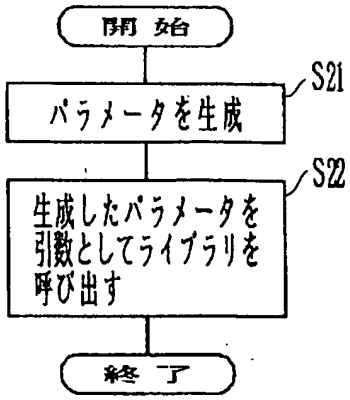
【図8】

9台のプロセッサによりサイクリック分割された
グローバル変数を示す図



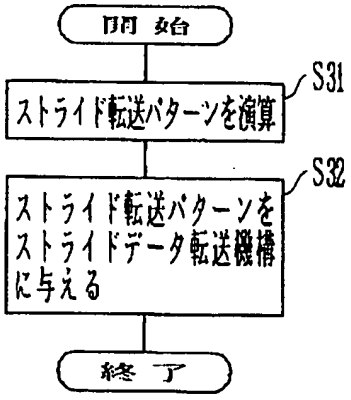
【図11】

プログラムによる処理のフローチャート



【図12】

ライブラリによる処理のフローチャート



【図９】

コンパイラによる処理のフローチャート（その１）